Argonne NATIONAL LABORATORY

Transitioning from Cobalt to PBS at the ALCF

Bill Allcock, ALCF Developer Sessions Webinar, April 30th, 2021

#include std_disclaimer.h

- “We all know that no plan survives contact with the enemy” - Helmuth von Moltke the Elder
- What I am going to present are our current plans and thoughts, however…
  - We have never run PBS in production
    - We don’t know how it will perform at scale
    - We have no experience tuning it
    - We don’t have years of user feedback
  - We are running on machines much different than we are used to
    - GPUs; We have experience with Theta-GPU and that is useful, but it is not with PBS.
- I am passing along what we have figured out so far. I am sure there are many things we don’t know.
- We will be flexible and responsive based on operational experience and user feedback.
- The gist here is:

THIS IS ALL SUBJECT TO CHANGE ☺

Tools and References to Aid in the Transition

- All page and section references refer to the 2020 guide (2021 just came out)
- Going from Cobalt to PBS is going to be sort of like going between Linux and AIX
  - You probably are going to know what command you want to use to do a certain thing.
  - But the differences in the command line options will drive you nuts…
  - So, what to do about that…
- One big advantage of this transition is the improvement in documentation. The PBS documentation is extensive and quite good:
  - https://www.altair.com/pbs-works-documentation/
  - The User's Guide and the Reference Guide are likely to be the two most relevant guides.
- There is a document we have produced that shows the mappings from Cobalt qsub command lines to PBS qsub command lines.
  - https://anl.box.com/s/394yay69qutg3ctpxav4719in7izgu4o
- There is also a CLI tool called `qsub2pbs` that is deployed on all our production systems that will take a Cobalt qsub command line and convert it to a PBS command line.
  - Just recall your `qsub` line and change qsub to `qsub2pbs`
  - If you want a script rather than a command line, add the `-directives` option

High level concepts

- You will hear two terms: OpenPBS and PBS Pro
  - OpenPBS is an open source project and we are contributing to it.
  - PBS Pro is OpenPBS with
    - More extensive hardening and testing
    - Paid priority support
    - Some optional features like an accounting system
    - They do start a PBS Pro release from the OpenPBS master branch
- Everything in PBS refers to a “vnode” or virtual node.
  - A vnode may be equivalent to a physical node, in the ALCF, it likely will be.
  - However, we could, for instance, take a ThetaGPU node (if it were running PBS) and make a physical node into 8 vnodes, each with 1 GPU, 1/8th of the cores, and 1/8th of the RAM.
  - I am going to use the terms nodes and vnodes interchangeably, but I will always be referring to vnodes unless I explicitly say otherwise.
- We are ***PLANNING*** on running a single PBS instance for the entire ALCF
  - remember that whole subject to change thing?
  - You can launch a job to any resource from any resource.
  - You can select multiple resources in a single job submission (more about that later)

qsub

- Has the same function as in Cobalt: it puts a job into a queue to be executed.
- The biggest difference is in resource selection
  - In Cobalt, since we ran a separate instance on every resource and our resources were primarily homogeneous, the resource was usually implied. You get Theta nodes on Theta, Cooley nodes on Cooley, etc..
  - Where there was heterogeneity, we normally handled that with a queue; big_mem queue for nodes with more memory, etc..
- In PBS, there is an “sql like” select syntax for selecting the resources you need.
  - Users Guide, Section 4.3, starting on UG-51
  - all selections are done with “minus lower case L”; With this font, that isn’t clear.
  - There are two types of resources, job wide and “chunks” (their term, don’t blame me)
    - wall time applies to the job, so is a job wide resources and is selected via `-l walltime=hh:mm:ss`
    - If we do have “one scheduler to rule them all”, then you might want 128 nodes on Polaris (an upcoming machine) and 8 on Cooley2 (our upcoming Cooley replacement) and that might look like `-l select 8:system=cooley2, 128:system=polaris`
    - There are a huge number of built-in resources in PBS. You can see them in the Reference Guide, section 5.6, RG-265. system is a “custom” resource we intend to create to make the selection easier.
    - You likely won’t use anything other than what I show up above. The selection is much more apropos if you have a heterogenous facility or nodes are shared and so you are looking for a node with sufficient resources left.
    - A minimal qsub:
      - `qsub -A project1 -l walltime=01:00:00 -l select 128:system=polaris - a.out`
      - You use the double dash “—” syntax when you want to run an executable directly

qsub differences continued…

- Table 2.1 in the Users Guide on page UG-24 is quite handy for figuring out qsub options.
- environment variables
  - Cobalt: –env <variable list>; PBS: –v <variable list> or –V to copy your entire environment
- Passing info to the server
  - Cobalt: --attrs pubnet; PBS: -l select 128:pubnet=True
  - Cobalt uses server attributes; in PBS you select custom resources
- Dependencies
  - Cobalt: --dependencies:<job list>; PBS: -W depend=afterok:<job list>
  - Section 6.2.1 in the users guide, UG-107 shows the alternatives to afterok
- Getting email
  - Again, PBS gives you more options, but it requires two options. Users Guide Section 2.5.1 UG-25
  - Cobalt: -M or –notify <addresses>; PBS: -M <addresses> –m <mail points> be (beginning and end) will get you equivalent functionality to Cobalt

qsub capabilities PBS has that Cobalt does not

- Job Arrays
  - If you need to run the same job many times on different inputs
  - Reduces the load on the scheduler
  - Every sub-job is treated as a job, so we believe priority will accrue normally, but we need to test.
  - How much interest is there in this feature?
  - Unless it has unexpected effects on the scheduling, we will likely enable this.
- Ability to re-run a job
  - If your job dies, can it be requeued and restarted without human intervention?
  - Basically the difference between your job being requeued or deleted on error.
  - Of how much interest is this?
  - The default is that jobs are re-runnable, but if the common case is that they are not, we will either change the default or if we will disable it completely.
  - Table 6.2 in the users guide, page UG-119
- Job history
  - Your job will remain visible in qstat and qselect for a period of time after it runs (default is two weeks)
- There are many more. See the PDF linked in the first slide or the PBS documentation for more details.

qsub capabilities Cobalt has that PBS does not

- --cwd
  - You can't change the directory from the CLI in PBS
  - in PBS use a script and do a cd in the script.
- --user_list
  - Cobalt allows the submitting user to authorize other users or a group to manipulate (hold, delete, etc) the job
  - PBS does not have this capability.
- --debuglog
  - PBS does not generate the equivalent of the .cobalt file.

- How much interest would there be in equivalent features? We can add features, but we have to prioritize.

Commands other than qsub

- qstat – displays status of jobs, queues, and servers
  - BTW, PBS qstat truncates output
  - If you want to either parse the data or see the whole thing, use the –json option.
- qalter – change the attributes of a queued job
- qdel – delete a queued job
- qhold – put a queued job on hold
- qmove – move a job from one queue to another
- qrls – releases a user hold on a job
- qselect – allows you to query for jobs based on many different job attributes.
- tracejob – extracts and prints log messages for a PBS job
- With the exception of qselect and tracejob, the other commands work very similarly to Cobalt.

Reservations

- Conceptually, they are similar to Cobalt
  - Our policies will likely be much the same, which is to say they are the exception, not the rule
  - And yes, PBS has the ability to let users set reservations, but we will be disabling that
    - I honestly have no idea how that would actually work…
- Reservations are the one place I think Cobalt is better
- Here are the primary differences you will notice
  - The queue is part of the reservation, not a separate entity that is bound to the reservation as in Cobalt
  - The queue goes away when the reservation ends
    - Any jobs still in the queue are lost
  - Any running jobs started during the reservation are killed when the reservation ends
    - There is no `releaseres` only `pbs_rdel`
- This is an area we intend to make contributions in the near future
  - We plan to allow for an optional queue to be passed in to the reservation
  - We plan to add an option to leave running jobs alone.
- There is an interesting option to turn a running job into a reservation
  - If you are debugging at scale, this could be useful, as you won't have to wait through the queue again
  - It could also be wasteful if large chunks of the machine sit idle while you debug.

Questions?

Thank You